НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___" _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та компоненти»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Програмні засоби розпізнавання показань лічильників енергоспоживання _____

Виконав (-ла):

студент (-ка) IV курсу, групи КВ-63
2(шифр групи)

Білих Олександр Сегійович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц. каф. СпіСКС, к. т. н., доцент Петрашенко А.В._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____2020 р.

ЗАВДАННЯ
на дипломний проєкт студента

Білих Олександра Сергійовича_____
(прізвище, ім'я, по батькові)

1. Тема проєкту «Програмні засоби розпізнавання показань лічильників енергоспоживання» _____
_____
_____ ,
керівник проєкту доц. каф. СпіСКС, к. т. н., доцент Петрашенко А.В. _____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «29» травня 2020 р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. Технічне завдання _____

4. Зміст пояснювальної записки _____
- Аналіз предметної області та існуючих рішень
- Опис та аналіз мов програмування, обгрунтування вибору мов та технологій для розробки _____
- Структурно-алгоритмічна організація_____
- Аналіз розроблених програмних засобів_____

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) _____
- Архітектура програмного засобу. Схема структурна _____
- Структура нейронної мережі. Схема структурна _____
- Послідовність підготовки зображення перед розпізнаванням. Схема алгоритму _____
- Алгоритм розпізнавання цифр. Схема алгоритму _____

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	к. т. н., доцент Клятченко Я.М.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Видача завдання на дипломне проєктування	01.12.19	
2	Аналіз існуючих рішень	25.01.20	
3	Вибір середовища та засобів розробки	10.02.20	
4	Розробка програмного засобу	10.03.20	
5	Відлагодження програмного засобу	25.03.20	
6	Підготовка пояснювальної записки	20.04.20	
7	Оформлення матеріалів проєкту	15.05.20	
8	Попередній розгляд дипломного проєкту	22.02.20	

Студент _____ О.С. БІЛИХ
(підпис)

Керівник проєкту _____ А.В. ПЕТРАШЕНКО
(підпис)

* Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Пояснювальна записка містить 55 сторінок, 3 таблиці, 16 рисунків, 17 джерел.

Об’єкт дослідження – програмні засоби розпізнавання показань лічильників енергоспоживання.

Предмет дослідження – методи розпізнавання символів на зображеннях.

Мета роботи – ознайомлення з відомими методам розпізнавання символів, в особливості цифр, на зображенні, дослідження їх можливостей та ефективності, а також використання для розробки програмного забезпечення для розпізнавання цифр на зображені лічильника енергоспоживання.

Метод дослідження – вивчення джерел інформації, аналіз відомих алгоритмів обробки зображення та методів розпізнавання символів на ньому, аналіз отриманих результатів і написання висновків.

В процесі розробки було проаналізоване використання мови Python 3, її фреймворків та бібліотек для обробки зображення (OpenCV), побудови та використання з її допомогою нейронної мережі для розпізнавання цифр на зображеннях (TensorFlow та Keras); використання існуючих наборів данних для навчання нейронної мережі.

Технічне завдання було розділене на структурні частини для розподілення роботи на етапи. Розроблена програма приймає на вхід зображення, на зміст якого вона була попередньо налаштована. Процес роботи програми розроблений таким чином, що вхідне зображення спочатку підлягає попередній обробці, сегментується, а далі на результатах сегментування нейронна мережа класифікує цифри. Всі етапи роботи в режимі налаштування фіксуються та зберігаються.

Ключові слова: характеристика зображення, обробка зображення, розпізнавання символу, модель нейронної мережі, сегментація, показання.

SUMMARY

The explanatory note contains 55 pages, 3 tables, 16 figures, 17 sources.

The object of research is a program for the recognition of the energy meter reading.

The subject of research - the method of recognition of symbols in the image.

The purpose of the work - familiar with the methods of symbols identifying, in particular numbers, in the image, researching their capabilities and effectiveness, using for developing software for recognizing numbers in a picture of an energy meter.

The research method is the study of information sources, analysis of known image processing algorithms and methods for recognizing characters on it, analysis of the results and writing conclusions.

During the development process, we analyzed the using of the Python 3 language, it's frameworks and libraries for image processing (OpenCV), building and using of a neural network for recognizing numbers in images (TensorFlow and Keras), using existing datasets for training a neural network.

The terms of reference were divided into structural parts in order to distribute the work into stages. The developed program accepts input images, the contents of which it was previously configured. The program operation process is designed in such a way that the input image is first pre-processed, segmented, and then the neural network classifies the numbers based on the segmentation results. All stages of work in the setting mode are observed and saved.

Keywords: image characteristics, image processing, character recognition, neural network model, segmentation, readings.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.002 ТЗ	Програмні засоби	4		
			розпізнавання показань			
			лічильників			
			енергоспоживання.			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Програмні засоби	2		
			розпізнавання показань			
			лічильників			
			енергоспоживання.			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045490.004 ПЗ	Програмні засоби	55		
			розпізнавання показань			
			лічильників			
			енергоспоживання.			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Білих О.С.			Програмні засоби розпізнавання показань лічильників енергоспоживання **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко А.В					
Консульт.						НТУУ «КПІ» ім. Ігоря Сікорського, ФПМ, КВ-63	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич К.В.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.005 Д1	Архітектура програмного	1		
			засобу.			
			Схема структурна			
	А4	ІАЛЦ.045490.006 Д2	Структура нейронної	1		
			мережі.			
			Схема структурна			
	А4	ІАЛЦ.045490.007 Д3	Послідовність підготовки	1		
			зображення перед			
			розпізнаванням.			
			Схема алгоритму			
	А4	ІАЛЦ.045490.008 Д4	Алгоритм розпізнавання	1		
			цифр.			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної			
			записки.			
			Графічний матеріал			

Зміст

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ. 2

2. ПІДСТАВА ДЛЯ РОЗРОБКИ. 2

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ. 2

4. ДЖЕРЕЛА РОЗРОБКИ. 2

5. ТЕХНІЧНІ ВИМОГИ. 2

5.1. Вимоги до програмного продукту, що розробляється. 2

5.2. Вимоги до апаратного забезпечення. 3

5.3. Вимоги до програмного та апаратного забезпечення користувача. 3

6. ЕТАПИ РОЗРОБКИ. 3

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Найменування роботи – дипломний проект на тему «Програмні засоби розпізнавання показань лічильника енергоспоживання».

Галузь застосування: автоматизація фіксації та надання показань лічильника енергоспоживання.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання проєкту бакалаврського рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ ПРОЄКТУ

Метою даного проекту є дослідження методів класифікації об'єктів на зображенні та розробка програмного забезпечення, що виконує розпізнавання цифр на вхідному зображенні показань лічильника.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

1. Вимоги до до програмного продукту, що розробляється:

1. можливість запуску на різних системах;
2. можливість зміни конфігурацій відповідно до умов;
3. можливість отримання покрокових результатів роботи програми;
4. можливість використання отриманих результатів після завершення роботи програми.

2. Вимоги до апаратного забезпечення:

- Оперативна пам’ять: 2 Гб;
- Процесор Intel Pentium G2030 і вище.

3. Вимоги до програмного та апаратного забезпечення користувача:

- операційна система, яка підтримує Python 3 та пакет бібліотек, що використовуються в програмі.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Видача завдання на дипломне проєктування	01.12.19	
2	Аналіз існуючих рішень	25.01.20	
3	Вибір середовища та засобів розробки	10.02.20	
4	Розробка програмного засобу	10.03.20	
5	Відлагодження програмного засобу	25.03.20	
6	Підготовка пояснювальної записки	20.04.20	
7	Оформлення матеріалів проєкту	15.05.20	
8	Попередній розгляд дипломного проєкту	22.02.20	

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.004 ПЗ	Програмні засоби	55		
			розпізнавання показань			
			лічильників			
			енергоспоживання			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Архітектура програмного	1		
			засобу.			
			Схема структурна			
	А4	ІАЛЦ.045490.006 Д2	Структура нейронної	1		
			мережі.			
			Схема структурна			
	А4	ІАЛЦ.045490.007 Д3	Послідовність підготовки	1		
			Зображення перед			
			розпізнаванням.			
			Схема алгоритму			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Білих О.С.			Програмні засоби розпізнавання показань лічильників енергоспоживання **Відомість технічного проєкту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко А.В.					
Консульт.						НТУУ «КПІ» ім. Ігоря Сікорського, ФПМ, КВ-63	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич К.В.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.008 Д4	Алгоритм розпізнавання	1		
			цифр			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП	Арк. 2

Пояснювальна записка
до дипломного проєкту

на тему: Програмні засоби розпізнавання показань лічильників енергоспоживання

Київ – 2020 року

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 4
ВСТУП 5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ 7
1.1 Лічильники енергоспоживання та аналіз процесу фіксування показань 7
1.2 Аналіз існуючих систем та програмних рішень 9
1.2.1 Онлайн кабінети для відправки показань лічильника 10
1.2.2 АСКОЕ 10
1.2.3 Tesseract 11
1.3 Висновок до розділу та аналіз вимог до функціональності 12
2 ОПИС ТА АНАЛІЗ МОВ ПРОГРАМУВАННЯ, ОБҐРУНТУВАННЯ ВИБОРУ МОВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ 14
2.1 Обґрунтування вибору мови програмування 14
2.1.1 Python 3 14
2.1.2 C/C++ 15
2.1.3 Аналіз мов 16
2.2 Бібліотеки для обробки та підготовки зображення 16
2.2.1 NumPy 17
2.2.2 OpenCV-Python 17
2.3 Інструменти для побудови нейронної мережі 18
2.3.1 TensorFlow 18
2.3.2 Keras 19
2.4 Огляд набору даних MNIST 19
2.5 Інструменти машинного навчання 20

2.5.1 Багатошаровий персептрон 20
2.5.2 Згорткова нейронна мережа 22
2.6 Аналіз способів розпізнавання символів 24
2.6.1 Ознакові методи 24
2.6.2 Нейронні мережі та глибоке навчання 25
2.7 Способи збереження даних та звернення до них 26
2.7.1 База даних та СУБД 26
2.7.2 CSV файл та стандартний CSV-модуль Python 26
2.8 Висновок до розділу 27
3 СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ 28
3.1 Структура програмних засобів 28
3.2 Підготовка вхідного зображення до наступних алгоритмів 28
3.3 Алгоритм знаходження циферблату 33
3.4 Алгоритм сегментації цифр на циферблаті 34
3.5 Алгоритм розпізнавання цифр 34
3.5.1 Розпізнавання нейронною мережею 35
3.5.2 Створення моделі 35
3.5.3 Навчання моделі 38
3.5.4 Використання моделі для класифікації 39
3.6 Алгоритми перевірки відповідей та їх запису 40
3.7 Прибирання залишків роботи програми 42
3.8 Висновок до розділу 43
4 АНАЛІЗ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ 45
4.1 Тестування системи 45
4.2 Аналіз отриманих результатів тестування 48
4.3 Рекомендації щодо використання розробки 48
4.4 Рекомендації щодо подальшого вдосконалення 50
4.5 Висновок до розділу 51
ВИСНОВКИ 52

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 54
ДОДАТКИ
ДОДАТОК А. ІАЛЦ045490.005 Д1. Архітектура програмного засобу. Схема структурна
ДОДАТОК Б. ІАЛЦ045490.005 Д2. Структура нейронної мережі. Схема структурна
ДОДАТОК В. ІАЛЦ045490.005 Д3. Послідовність підготовки зображення перед розпізнаванням. Схема алгоритму
ДОДАТОК Г. ІАЛЦ045490.005 Д4. Алгоритм розпізнавання цифр. Схема алгоритму
ДОДАТОК Д. Лістинг програми

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Датасет (англ. Dataset) – набір даних, найчастіше відповідає змісту таблиці бази даних, або статистичній матриці даних тощо.

Діджиталізація - зміни в усіх сферах суспільного життя, пов’язанні з використанням цифрових технологій.

Згорткова нейронна мережа (англ. Convolutional neural network, CNN) – спеціальна архітектура штучних нейронних мереж, що націлена на ефективне розпізнавання образів та входить в склад технологій глибокого навчання.

Інтерпретатор – програма чи технічні засоби, необхідні для виконання інших програм.

Компіляція – трансляція вихідного коду програми в бінарний код.

Ком’юніті (англ. Community) – спілка. В даному контексті мається на увазі круг людей, що використовують якісь певні технології та діляться між собою досягненнями, навичками, знаннями, порадами, тощо.

Конфігурація – сукупність налаштувань програми, що задається користувачем, а також процес зміни цих налаштувань відповідно до потреб користувача.

Кросплатформність – властивість програмного забезпечення працювати більш ніж на одній програмній або апаратній платформі.

Обробка зображень – процес аналізу та роботи з цифровим зображенням, направлений на покращення якості картинки чи вилучення інформації для подальшого її використання.

ОС – операційна система.

ПК – персональний комп’ютер

Скрипт – послідовність команд.

Фреймворк – заготовки, шаблони для програмної платформи, що визначають архітектуру програмної системи.

API (англ. Application programmin interface, укр. Програмний інтерфейс додатку) – набір засобів (класів, функцій, методів, процедур), завдяки яким одна комп’ютерна програма може взаємодіяти з іншою.

Deep Learning - (укр. Глибинне навчання) - це галузь машинного навчання, що ґрунтується на наборі алгоритмів, які намагаються моделювати високорівневі абстракції в даних, застосовуючи глибинний граф із декількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень

Machine Learning (ML, укр. машинне навчання) – набір алгоритмів та статистичних моделей, які використовуються в комп’ютерних системах для ефективного виконання певних типів завдань без використання явних інструкцій, натомість покладаючись на виявлення зв’язків та шаблонів у даних.

OCR (англ. Optical character recognition, укр. оптичне розпізнавання символів) – переведення зображення, що містить будь-який текст, у текстові дані.

ВСТУП

Тенденцій розвитку сучасного суспільства та технологій ведуть до того, що все більше тих чи інших професій автоматизуються. В цей же час, кількість людей на планеті збільшується кожен рік все швидше й швидше. Тож, виходячи з цього, росте й кількість як житлових об’єктів, так і кількість промислових, офісних, тощо. І кожен з них, потребує десятки електролічильників.

Не дивлячись на те, що лічильники стають все більш зручними та новітні їх версії мають можливість фіксування показань завдяки безпосередньому підключенню до них автоматизованих систем, вони водночас стають і дорожчими та складнішими у використанні, через що користувачі не завжди бачать сенс в переході на найновіші лічильники. Тож кожного місяця користувачі електроенергії зобов'язані надавати показання використаної електроенергії, попередньо переписавши їх на листок, а контролери показань лічильників кожного робочого дня обходять сотні об’єктів для фіксування показань. Тож, попри діджиталізацію, процес фіксування та надавання показань лічильників енергоспоживання особливо не змінився, як і кількість можливих помилок та шахраїв.

Звертаючи увагу на вищесказане та на те, що лічильники енергоспоживання намагаються максимально зручно групувати, на сьогоднішній день є актуальною задача створення системи, що автоматизує фіксування показань індукційних електролічильників.

Данна робота присвячена розробці додатку, який спрощує процес фіксування показань лічильників, використовуючи технології оптичного розпізнавання цифр.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Лічильники енергоспоживання та аналіз процесу фіксування показань

Завдяки технологічному прогресу, якість, технологічність та зручність використання лічильників теж покращуються. На сьогоднішній день використовується два види лічильників, кожен з яких має свої переваги та недоліки:

1) індукційний лічильник електроенергії (рисунок 1)

Переваги:

- не схильні до впливу стрибків у мережі або зниження напруги;
- надійні в експлуатації і мають тривалий термін служби;
- мають порівняно низьку вартість.

Недоліки:

- розрахунок електроенергії за звичайним (однозонним тарифом);
- ведення обліку електроенергії тільки в одному напрямку;
- практично повна відсутність захисту від розкрадання електроенергії;
- не може бути використаний в системах АСКОЕ (див. у пункті 1.2.2).

2) електронний лічильник електроенергії (рисунок 1)

Переваги:

- наявність декількох тарифів (2- і 3-зонний облік);
- ведення обліку у двох напрямках;
- фіксація несанкціонованого доступу в разі розкрадання електроенергії;
- зберігання даних з обліку електроенергії;
- можливість використання в системах АСКОЕ;

Недоліки:

- чутливість до перенапруження, стрибків у мережі та зниження напруги;
- у разі несправності зазвичай вимагає складнішого ремонту.

Рисунок 2 - Індукційний та електронний лічильники електроенергії.

Споживачі електричної енергії зобов'язані оплачувати в повному обсязі витрачену енергію. Для цього вони повинні передати показання лічильника електроенергії в відповідну інстанцію або ж виконати самостійний розрахунок енергоспоживання.

Прилади індукційного типу оснащені блоком-колесом з цифрами, розміщеним під рамкою. Воно відображає дані, необхідні для виконання розрахунків і передачі показань. Від моделі пристрою і кількості цифрових значень залежить, які показання лічильників електроенергії потрібно передавати. Найчастіше табло лічильників індукційного типу відображає від 5 до 7 цифр. Остання цифра виділяється із загального числа за рахунок різниці в розмірі, кольорі або виділення коми. У рідкісних випадках можуть бути виділені два останніх числа. При знятті показань лічильника електроенергії не враховуються числові значення, що йдуть після коми. Ці дані відображають соті і десяті частки кіловата, тому до уваги не беруться (рисунок 2). Існують модифікації пристроїв, де кома або інший вид виділення відсутня. У такій ситуації для проведення розрахунків беруться всі числові

значення, відображені на табло, в іншому випадку виникнуть розбіжності в оплаті, які все одно доведеться покривати.

Після установки або заміни електролічильника власнику видається акт, який підтверджує правильне проведення даної процедури. У документі фіксуються початкові цифрові значення. Щоб зняти дані з агрегату, потрібно перенести на папір всі числа, які відображаються приладом на даний момент, не враховуючи цифри після коми. Також не беруться до уваги нулі, що стоять до першого значущого числа, тобто 1 і більше.

Для проведення розрахунків знадобляться дані попереднього місяця. У перший місяць після установки лічильника, ці цифри беруться з акту. Далі доведеться вести журнал обліку або зберігати всі квитанції, щоб фіксувати показання.

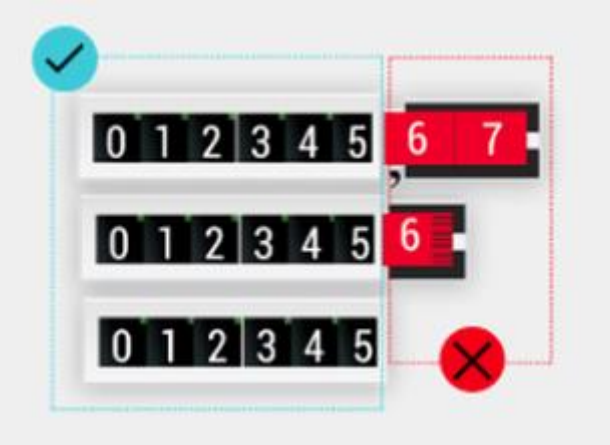


Рисунок 2 – Найпоширеніші види табло лічильників електроенергії.

1.2 Аналіз існуючих систем та програмних рішень

Існуючі засоби автоматизації надання показань лічильників, що пропонуються в нашій країні, орієнтовані або на те, що користувач сам зафіксує показання та надішле їх, або на те, що він встановить сучасні лічильники, що дозволяють використовувати автоматизовані системи, що працюють безпосередньо з лічильником.

1.2.1 Онлайн кабінети для відправки показань лічильника

Компанії постачальники електроенергії та надання послуг впроваджують електронні кабінети, в яких можна щомісячно надавати показання лічильників електроенергії та можливість передивлятись свої попередні дані.

Зазвичай, для цього потрібно одноразово зареєструватися, використовуючи паспортні дані та дані договорів з надання послуг електропостачання, а далі щомісяця:

-виконувати авторизацію;

-записувати показання лічильника;

-заносити показання у форми на сайті;

Для уникнення ситуацій з надання не вірних показань, постачальники використовують контролерів, що час від часу перевіряють ваші показання, та порівняння показань використання електроенергії (чи показання не менше за попередній період, чи не значно відрізняються обсяги місячного використання)

1.2.2 АСКОЕ

АСКОЕ (автоматизована система комерційного обліку електроенергії) – це система, що дає змогу вимірювати, збирати, накопичувати, обробляти й показувати інформацію про обсяги та параметри споживання електричної енергії за окремий період часу в певному місці.

Система АСКОЕ в режимі реального часу передає дані про:

1) витрату електроенергії для кожного споживача;
2) наявність проблем на лініях: аварій, перевантажень мережі, втрат, несправностей приладів тощо.

Запровадження АСКОЕ – це фундамент для трансформації наявної системи в сучасну «розумну» систему електропостачання з використанням прогресивної технології збору інформації про виробництво та споживання

електричної енергії. Це дасть змогу підвищити результативність, надійність і економічний ефект виробництва, а також посприяє стабільності у виробництві та процесі розподілу електроенергії.

Впровадження системи АСКОЕ передбачає заміну звичайного лічильника на багатофункціональний (зонний) прилад обліку. У багатоквартирному житловому будинку систему встановлюють для всіх жителів одночасно. За допомогою повідомлень на дошках оголошень мешканців заздалегідь попереджають про те, що в певні дні буде проводитися заміна приладів обліку. Для цього потрібно в зазначений час надати їм доступ до вашого електролічильника для його заміни.

1.2.3 Tesseract

Tesseract, дуже популярний двигун OCR, був спочатку розроблений Hewlett Packard у 1980-х, а потім був відкритий у 2005 році. Google прийняв проєкт у 2006 році і з тих пір його спонсорує.

Tesseract може працювати дуже добре в контрольованих умовах. Але буде працювати досить погано, якщо є значна кількість шуму або зображення не буде попередньо оброблено та очищено перед застосуванням Tesseract.
Так само, як глибинне навчання вплинуло майже на кожен аспект комп'ютерного зору, воно торкнулося розпізнавання символів та розпізнавання рукописного тексту.

Моделям, що ґрунтуються на глибинному навчанні, вдалося отримати безпрецедентну точність розпізнавання тексту, що виходить далеко за рамки традиційних підходів вилучення ознак та машинного навчання. Це було лише питанням часу, поки Tesseract включив глибоку модель навчання для подальшого підвищення точності OCR - і насправді цей час настав. Останній випуск Tesseract (v4) підтримує OCR на основі глибинного навчання, що значно точніше. Сам базовий двигун OCR використовує мережу

довгострокової пам'яті (LSTM), свого роду рекурентну нейронну мережу (RNN) .

1.3 Висновок та аналіз вимог до функціональності

Аналізуючи можливі існуючі засоби, можна зробити висновки, що для кінцевих користувачів існуючі засоби:

- лише частково полегшують задачу;
- потребують додаткових коштів, підтримки інших суб’єктів у бажанні та необхідності перейти на більш сучасні технології та засоби. Тобто потребують складної інтеграції.

Отже, для того, аби максимально полегшити процес надання постачальнику електроенергії показань лічильника у випадку, коли підключено індукційний лічильник електроенергії, а також інтеграцію системи, необхідно, опираючись на існуючі засоби, створити нову, більш індивідуальну систему.

Логічним буду створити систему з ширококутної макрокамери; модулю, який буде приймати фотографії з камери та відправляти їх на сервер чи ПК завдяки тим чи іншим засобам (Wi-Fi, Bluetooth, Ethernet); програмного забезпечення, що розпізнає показання лічильника та надішле їх до контролюючого засобу, або повідомить про помилку.

Цей проєкт зосереджений на створенні програмного забезпечення, яке дозволить розпізнати показання лічильника та передати їх до наступного програмного компоненту (вчасності – записати до таблиці). Орієнтуючись на те, що воно є частиною більшої системи, на вхід програми буде подаватися ряд зображень, зроблених з однакових ракурсу, відстані та за подібних умов освітлення.

Після аналізу завдання сформовано наступні функціональні вимоги:

1) можливість запуску додатку на різних системах/платформах:

2) можливість тонкого налаштування під умови використання та міграції для вирішення схожих проблем:
3) наявність коментарів з однозначним поясненням виконання тих чи інших кроків біля компонентів програми, що можна налаштувати:
4) можливість покрокового відстеження роботи програми:
5) можливість використовувати отриманні результати після завершення роботи програми.

2 ОПИС ТА АНАЛІЗ МОВ ПРОГРАМУВАННЯ, ОБҐРУНТУВАННЯ ВИБОРУ МОВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ

2.1 Обґрунтування вибору мови програмування

Для роботи з нейронними мережами найпоширенішими за 2019 рік є мови програмування Python (тут і в подальших текстах мається на увазі третя версія мови, тобто Python 3) та C/C++; для обробки зображень – C#, C/C++, Python та інші.

2.1.1 Python

Python – це високорівнева, інтерпретуєма та універсальна мова динамічного програмування, що сфокусована на читабельність коду та швидкість розробки. Синтаксис мови Python дозволяє програмістам писати програми з меншою кількістю коду, в порівнянні з тими ж Java чи C++. Мова включає в себе достатньо обширу кількість стандартних бібліотек.

Характерними особливостями мови є те, що вона:

- інтерпретуєма (хоча є можливість також компілювати код завдяки додатковим інструментам)
- інтерактивна
- модульна
- динамічна
- націлена на об’єктно-орієнтоване програмування
- має засоби функціонального програмування
- високорівнева
- розширювана в C/C++, Java
- включає в себе автоматичну збірку сміття
- простота та лаконічна

Програмні засоби, створені на цій мові:

- кросплатформні
- зручні для інтеграції в інші проєкти

Зазначені вище факти призвели до того, що мова зібрала досить велику кількість прихильників, що в подальшому дозволило ще більше розширити її потужність за рахунок створених на її основі бібліотек, фреймворків та готових компонентів.

На сьогоднішній день мову частіше використовують для створення web-додатків та системних скриптів, машинного та глибокого навчання, роботи з великими даними та їх візуалізації.

2.1.2 C/C++

Мова програмування С спочатку була розроблена і реалізована в операційній системі UNIX Деннісом Рітчі. Мова С не прив'язана до якогось конкретного обладнання або системі. Це полегшує користувачеві написання програм, які будуть працювати без багатьох (або хоча з мінімальними) змінами практично на всіх машинах.

Мова С часто називають комп'ютерною мовою середнього рівня, оскільки вона поєднує в собі елементи мов високого рівня з функціональністю асемблера. Програмування на С дозволяє маніпулювати бітами, байтами і адресами, надаючи програмісту більший контроль над тим, як буде вести себе програма, і більш прямий доступ до механіки базового обладнання.

С++ - це розширена версія мови С. С++ включає в себе все, що є частиною С, і додає підтримку об'єктно-орієнтованого програмування (ООП). Крім того, С++ також містить безліч поліпшень і функцій, які роблять його «кращим С» навіть не через можливість використання ООП.

2.1.3 Аналіз мов

Аналізуючи дані, ми бачимо, що найбільш універсальними є мови C/C++ та Python. В основному, це досягається завдяки великій кількості бібліотек для обох мов, а також велика кількість готових інструментів та широке ком'юніті. Тож раціональніше за все обирати між ними.

В даному випадку, ми можемо орієнтуватися на два фактори: швидкість розробки та швидкість роботи програми. Опираючись на загальнодоступні дослідження та характеристики мов, можна зробити висновки, що при потребі в швидкодії програмного забезпечення, краще звертати увагу на C/C++, проте для цього потрібно водночас мати навички та час для оптимізації коду. З другого боку, Python дозволяє пришвидшити розробку та зменшити кількість коду до мінімуму завдяки лаконічному синтаксису, зручно написаним бібліотекам та їх простій інтеграції у код.

Беручи до уваги те, що сказано вище, а саме:

- необхідність інтегрування програми в інші;
- можливість розширення функціоналу;
- необхідність швидкого та гнучкого налаштування;

було обрано мову програмування Python.

2.2 Бібліотеки для обробки та підготовки зображення

Загальні завдання зводяться до зчитування зображення, представлення його у зручній до обробки форми та виконання основних операцій (кадрування, відображення, обертання, сегментація, класифікація, витяг ознак, відновлення і розпізнавання).

2.2.1 NumPy

NumPy - це одна з основних Python-бібліотек з підтримкою масивів.

Основним об'єктом NumPy є однорідний багатовимірний масив (в numpy називається numpy.ndarray). Це багатовимірний масив елементів (зазвичай чисел), одного типу.

Найбільш важливі атрибути об'єктів ndarray:

- ndarray.ndim - число вимірювань (частіше їх називають "осі") масиву.
- ndarray.shape - розміри масиву, його форма. Це кортеж натуральних чисел, що показує довжину масиву по кожній осі. Для матриці з n рядків і m стовпів, shape буде (n, m). Число елементів кортежу shape одно ndim.
- ndarray.size - кількість елементів масиву. Очевидно, дорівнює добутку всіх елементів атрибута shape.
- ndarray.dtype - об'єкт, що описує тип елементів масиву. Можна визначити dtype, використовуючи стандартні типи даних Python. NumPy тут надає цілий букет можливостей, як вбудованих, наприклад: bool_, character, int8, int16, int32, int64, float8, float16, float32, float64, complex64, object_, так і можливість визначити власні типи даних, в тому числі і складові.
- ndarray.itemsize - розмір кожного елемента масиву в байтах.
- ndarray.data - буфер, який містить фактичні елементи масиву. Зазвичай не потрібно використовувати цей атрибут, так як звертатися до елементів масиву найпростіше за допомогою індексів.

Зображення переводиться у вигляд стандартного масиву NumPy. Таким чином, при виконанні основних NumPy-операцій (зрізи, маски, примхлива індексування) ми можемо змінювати піксельні значення зображення.

2.2.2 OpenCV-Python

OpenCV (Open Source Computer Vision Library) - одна з найпопулярніших бібліотек для додатків з комп'ютерного зору. OpenCV-Python - це Python-версія інтерфейсу для OpenCV. Наявність коду на C / C ++

в бек-енд гарантує швидкість бібліотеки, а Python-обгортка у фронт-енді забезпечує легкість настройки і розгортання. Завдяки цьому OpenCV-Python є відмінним рішенням для високонавантажених обчислювальних програм з комп'ютерного зору.

Усі структури масивів OpenCV перетворюються на масиви Numpy і з них. Отже, які б операції ви не могли робити в Numpy, ви можете комбінувати їх з OpenCV, що збільшує кількість зброї у вашому арсеналі. Крім цього, з цим можуть використовуватися кілька інших бібліотек, як SciPy, Matplotlib, який підтримує Numpy.

2.3 Інструменти для побудови нейронної мережі

На сьогоднішній день усі найбільші фреймворки та бібліотеки машинного та глибинного навчання підтримують мову Пайтон. Найбільш популярні з них TensorFlow, Keras (зазвичай використовується як front-end обгортка для TensorFlow), та PyTorch.

2.3.1 TensorFlow

TensorFlow - це фреймворк від Google із відкритим кодом для глибокого навчання. Він надає потужні можливості для структуризації наборів даних, роботи з тензорами (багатовимірні масиви, базовий будівельний блок нейронних мереж) та побудови архітектури глибокого навчання.

Ми можемо безпосередньо працювати з TensorFlow, щоб створити нові алгоритми нейронної мережі та тонко налаштувати моделі нейронної мережі.

При бажанні працювати зі стандартними моделями нейронних мереж, можна використовувати Keras як фронт-енд частину до TensorFlow.

2.3.2 Keras

Keras - бібліотека для глибокого навчання з відкритим кодом. Вона дозволяє швидко експериментувати, надаючи розробникам доступ до стандартних моделей нейронної мережі за допомогою простої моделі програмування.

Keras - це система глибокого навчання на високому рівні, яка працює над TensorFlow, Microsoft Cognitive Toolkit або Theano (але на практиці найчастіше використовується з TensorFlow). Keras пропонує зручні програмні абстракції, які дозволяють вам працювати з конструкціями глибокого навчання, такими як моделі, шари та гіперпараметри, а не з тензорами та матрицями.

2.4 Огляд набору даних MNIST.

Набір даних MNIST – загальнодоступний набір рукописних цифр. Складається з навчального набору з 60000 прикладів і тестового набору з 10 000 прикладів. Цифри в наборі нормалізовані за розміром та розташовані в центрі зображення. Розмір зображень фіксований та становить 28*28 пікселів

MNIST було побудована з набору NIST (National Institute of Standards and Technology), в якому кожний з сегментованих символів займає растровий діапазон 128*128 пікселів і позначений одним з 62 класів: “0”-“9”, “A”- “Z” та “a”-“z”. Набір зображень у базі даних MNIST - це поєднання двох оригінальних наборів даних NIST: Special Database 1 та Special Database 3, які складаються з цифр, записаних відповідно старшокласниками та працівниками Бюро перепису населення США. Набір навчальних тренувань у розмірі 60 000 містить приклади приблизно 250 письменників. Навчальний набір та тестовий набір не суперечать друг другу. Приклад даних набору зображено на рисунку 3.

За допомогою цього навчального та тестового наборів було перевірено багато методів та моделей. Приклад

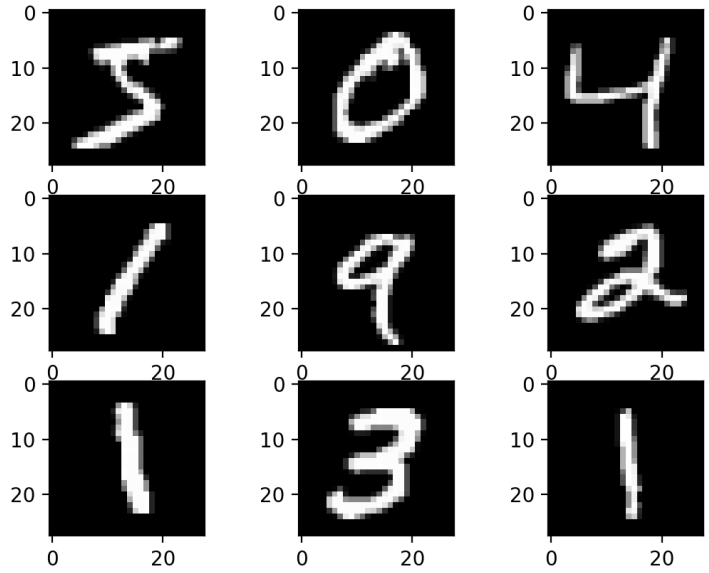


Рисунок 3 – Приклад рукописних цифр набору MNIST.

2.5 Інструменти машинного навчання

2.5.1 Багатошаровий персептрон (MLP)

Персептрон - це бінарний алгоритм класифікації, змодельований після функціонізації людського мозку - він був призначений для імітації нейрона. Персептрон, не зважаючи на просту структуру, має здатність вчитися і вирішувати дуже складні проблеми.

Багатошаровий персептрон (MLP) - це група персептронів, організованих у кілька шарів, які можуть точніше відповісти на складні запитання. Кожен персептрон першого шару (вхідного) посилає сигнали всім персептронам у другому шарі (прихованого), і так до останнього (вихідного). MLP містить вхідний шар, принаймні один прихований шар, і вихідний шар (рисунок 4).

Персептрон навчається наступним чином:

1) бере вхідні дані, які подаються у вхідному шарі, множить їх на їх ваги та обчислює суму;

2) сумує перший шар, помножений на "вагу зміщення". Це технічний крок, який дає можливість перемістити вихідну функцію кожного персептрона (функція активації) вгору, вниз, вліво та вправо на графіку чисел;
3) подає суму за допомогою функції активації - у простій системі персептрон функція активації є кроковою функцією;
4) результатом крокової функції є вихід.

Додаючи наступні кілька інгредієнтів, архітектура персептрону стає повноцінною системою глибокого навчання:

- Функції активації та інші гіперпараметри - повна нейронна мережа використовує різноманітні функції активації, які виводять реальні значення, а не булеві значення, як у класичному персептроні. Він є більш гнучким щодо інших деталей навчального процесу, таких як кількість навчальних ітерацій (ітерацій та епох), схеми ініціалізації ваги, регуляризації тощо. Все це можна настроїти як гіперпараметри.
- Зворотне розповсюдження - повна нейронна мережа використовує алгоритм зворотного розповсюдження для виконання ітеративних зворотних проходів, які намагаються знайти оптимальні значення ваги персептрона, щоб генерувати найбільш точний прогноз.
- Розширена архітектура - повні нейронні мережі можуть мати різноманітну архітектуру, яка може допомогти вирішити конкретні проблеми. Кілька прикладів - рекурентні нейронні мережі (RNN), згорткові нейронні мережі (CNN) та генеративні змагальні мережі (GAN) .

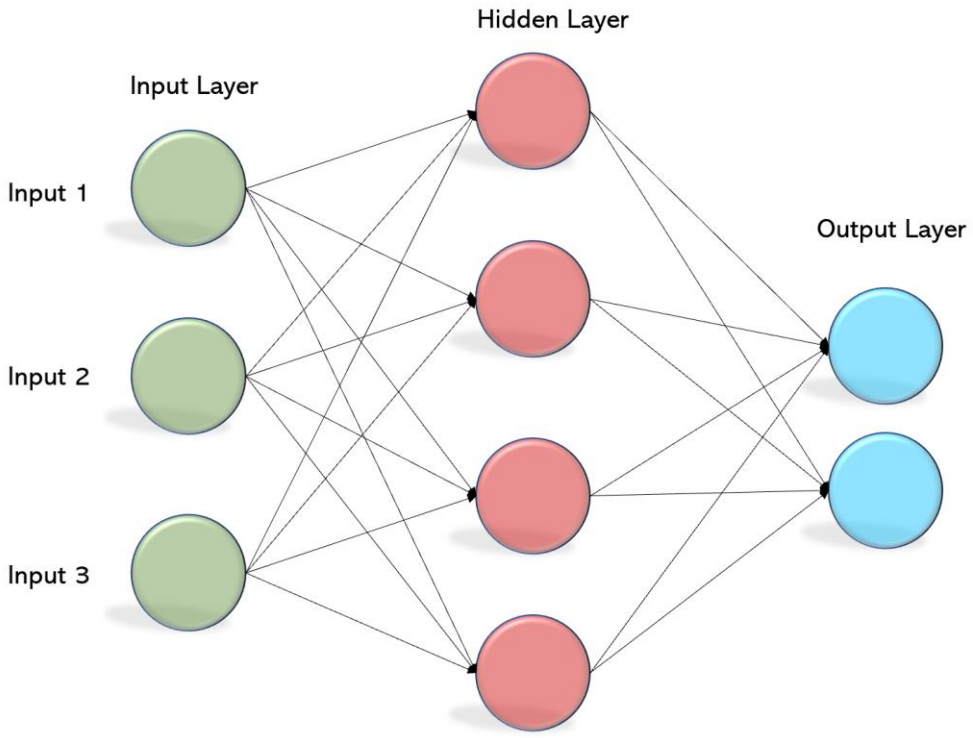


Рисунок 4 – Структура багатошарового персептрону.

2.5.2 Згорткова нейронна мережа (CNN)

CNN є багатошаровою нейронною мережею, що використовується для аналізу та класифікації зображень, сегментації або виявлення об'єкта. CNN працюють над тим, щоб зменшити зображення до його основних характеристик і використовувати комбіновані ймовірності виявлених ознак, що з'являються разом для визначення класифікації. Однією з переваг CNN в порівнянні з іншими алгоритмами класифікації є те, що їм потрібно менше гіперпараметрів і менший нагляд.

CNN складаються декількох типів шару: згортки (Convolution), вибірки (Max-Pooling), об'єднання (Flatten) та повністю з'єднані (Fully-Connected). В бібліотеці Keras їм відповідають класи Conv2D, MaxPooling2D, Flatten та Dense, відповідно.

У згорткових шарах вхід аналізується набором фільтрів, які виводять мапу функції (рисунок 5).

Рисунок 5 – Схематичне зображення роботи шару згортки.

Потім цей вихід надсилається на шар вибірки, що зменшує розмір карти функцій (рисунок 6). Це допомагає скоротити час обробки, конденсуючи карту, до найважливішої інформації.

Рисунок 6 – Схематичне зображення роботи шару вибірки.

Процеси згортки та вибірки повторюються кілька разів, з кількістю повторень, залежно від мережі, після чого виводи конденсованих функцій карти надсилаються до серії шарів FC. Потім ці шари FC згладжують карти разом і порівнюють ймовірності кожної ознаки, що виникають спільно з іншими, до тих пір, поки не буде визначена найкраща класифікація. Рисунок 7 демонструє наочно структуру згорткової мережі, в тому числі послідовність дії до моменту отримання результату.

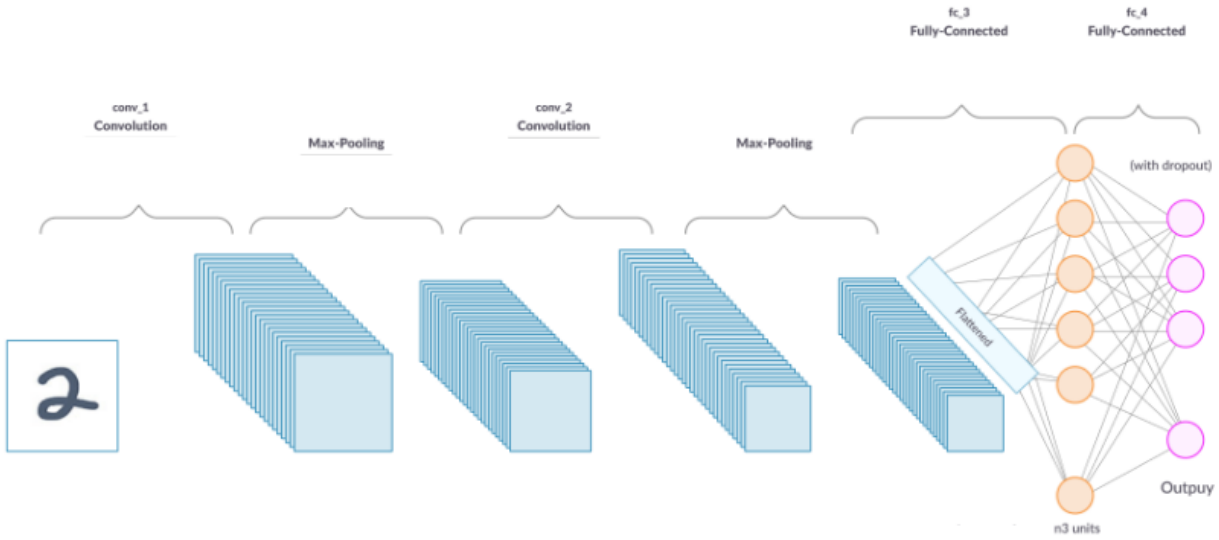


Рисунок 7 – Архітектура згорткової мережі (CNN).

Ця архітектура дозволяє CNN дізнаватися положення та масштаб ознак у різних зображеннях, що робить її особливо корисною та показовою при класифікації ієрархічних чи просторових даних та вилученні немаркованих ознак. На жаль, ця структура вимагає, щоб CNN приймали лише вводи фіксованого розміру, і це дозволяє їм надавати виходи лише фіксованого розміру.

2.6 Аналіз способів розпізнавання символів

2.6.1 Ознакові методи

Не дивлячись на те, що назви “механічних” методів розпізнавання символів з роками постійно змінюються та групуються під різними іменами, досить однозначно можна сказати, що для них характерним є використання тих чи інших ознак зображення та зіставлення їх до шаблонних.

Прикладами є такі методи, що базуються на:

- порівнянні усіх точок вхідного зображення з усіма точками шаблонних зображень, щоб знайти шаблон, який має найбільшу кількість збігів;

- зіставленні топології символу на зображенні до еталонів, тобто орієнтування на структуру символу;

У першому випадку, метод досить стійкі до дефектів зображення і має високу швидкість обробки вхідних даних, але надійно розпізнає тільки ті шрифти, шаблони яких йому „відомі”. І якщо розпізнаний шрифт хоч трохи відрізняється від еталонного, метод може робити помилки навіть при обробці дуже якісних зображень.

В другому маємо зворотні переваги та недоліки. Перевага методу – стійкість до зсуву і повороту символу на невеликий кут, до різних стильових варіацій шрифтів. Однак, при повороті на кут, більший за якусь межу, даний метод не може бути використаний для розпізнавання символів. При застосування цього методу неважливими стають такі ознаки як розмір символу, що розпізнається і навіть шрифт, яким його надруковано. Проте, основною проблемою цього методу є ідентифікація знаків, які містять певні дефекти (наприклад, розрив ліній або з'єднання сусідніх ліній).

2.6.2 Нейронні мережі та глибоке навчання

Нейронні мережі класифікують, передаючи вхідні значення через ряд нейронних шарів, які виконують складні перетворення на даних.

Сильні сторони: нейронні мережі дуже ефективні для проблем з високою розмірністю або при складних зв'язках між змінними. Наприклад, нейронні мережі можна використовувати для класифікації та маркування зображень, аудіо та відео, проведення аналізу настроїв на тексті та класифікації інцидентів безпеки на категорії ризику.

Слабкі сторони: Нейронні мережі теоретично складні, їх важко реалізувати, вони вимагають ретельного налаштування та обчислювальної інтенсивності. Малий досвід в глибинному навчанні може коштувати втраті великої кількості часу, що було задіяно на розробку моделі, її навчанні та

тестуванні. Не завжди складніші моделі є оптимальним шляхом для вирішення складнішої на перший погляд людини задачі.

2.7 Способи збереження даних та звернення до них

2.7.1 База даних та СУБД

База даних - це організований набір даних, який, як правило, зберігається та доступний в електронному вигляді з комп'ютерної системи. Там, де бази даних складніші, вони часто розробляються з використанням формальних методів проєктування та моделювання .

Система управління базами даних (СУБД) - це програмне забезпечення, яке взаємодіє з кінцевими користувачами , програмами та самою базою даних для збору та аналізу даних. Програмне забезпечення СУБД додатково охоплює основні засоби, наданні для адміністрування бази даних. Загальна сума бази даних, СУБД та пов'язаних з ними програм може називатися "системою баз даних". Часто термін "база даних" також використовується для посилання на будь-яку СУБД, систему бази даних або додаток, пов'язаний з базою даних.

2.7.2 CSV файл та стандартний CSV-модуль Python 3

CSV (від англ. comma-separated values ‘значення, розділені комою’, іноді - character-separated values ‘значення, розділені символом’) — файловий формат, котрий є відмежовуваним форматом для представлення табличних даних, у якому поля відокремлюються символом коми та переходу на новий рядок. Поля, що містять коми, декілька рядків, або лапки (позначаються подвійними лапками), мають обмежуватися з обох боків лапками.

Модуль CSV дає можливість розробнику можливість виконувати структурний аналіз файлів CSV. Не дивлячись на те, що формат не стандартизовано, більшість файлів, що його використовують, схожі та дотримуються однакових правил. Тож модуль може розпізнати більшу

частину цих файлів. Звісно, програміст може як зчитувати інформацію з файлу, так і записувати її.

Формат зручно використовувати, якщо потрібно зберігати структурно прості дані та мати можливість ознайомлення з ними поза програми.

2.8 Висновок до розділу.

Від вибору технології для розробки залежить те, якого вигляду набуває програмне забезпечення, на скільки швидко воно працює та на скільки зручно його розробляти та надалі використовувати, який набір інструментів буде доступним до використання.

Мовою розробки програмного засобу обрано Python 3. Вибір засобів розробки робиться на основі функціональних та нефункціональних вимог. Перелік інструментів та методів підходу до рішення задачі, що зазначений в цьому розділі, є щонайменше достатнім для покриття вимог до програмного засобу, і, що не менш важливо, зручним для мене у використанні. При бажанні чи необхідності розширення використовуваних засобів, можна вільно додавати їх, що робить програму більш гнучкою та сприятливою до розширення.

3 СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ

3.1 Структура програмних засобів

Загальна структура програмного забезпечення проєкту поділяється на такі основні частини:

1) отримання програмою зображення;
2) підготовка (обробка) зображення;
3) виявлення на зображенні табло;
4) виокремлення цифр на табло;
5) розпізнавання цифр;
6) отримання попередніх показань з таблиці;
7) виділення помилок;
8) запис показань до таблиці;

В цьому проєкті ми зацікавлені в розробці пунктів 1-5, проте для наочності реалізації та можливості додаткової перевірки отриманих результатів, доцільно буде розробити повний програмний комплекс.

3.2 Підготовка вхідного зображення до наступних алгоритмів

Беручи до уваги, що система, яка буде робити знімки, буде розміщена статично, то є сенс одразу відсікати частини зображення, що нас не цікавлять, без використання додаткових алгоритмів.

Алгоритм поєднує у собі наступну послідовність дій:

1) відкриття зображення

Метод cv2.imread(‘path_to_file’, (flag)) дозволяє відкрити зображення ‘path_to_file’ у виді тримірного NumPy масиву (numpy.ndarray [row(height)][column(width)][color(RGB)]).

2) зміна розміру

Функція def image_resize(image, image_width=0, image_height=0))
на основі бібліотечної функції cv2.resize() виконує зміну розміру зображення.

image - параметр функції, що приймає зображення у вигляді масиву, отримане в попередньому пункті. Він є обов'язковим.

image_width – параметр, в якому вказується бажана ширина зображення. Він є обов'язковим. Може бути нуль чи будь-яке натуральне число.

image_height – параметр, в якому вказується бажана висота. Може бути не використаний, набувати значення нуль або настуральне.

Обов'язковим є те, щоб хоч б одне з двох значень (ширина чи висота) було не нуль. Тобто, наприклад, image_resize(image, 0, 250) чи image_resize(image, 250, 0) = image_resize(image, 250). У випадку, якщо один з параметрів 0, його величина обчислюється пропорційно співвідношенню бажаного другого параметру до його первинної величини.

3) виокремлення необхідної частини

image[y1:y2, x1:x2], де y1 та y2 – початкова та кінцева точки вертикалі необхідної частини зображення, відповідно; x1 та x2 - початкова та кінцева точки горизонталі необхідної частини зображення (рисунок 8), відповідно.

Рисунок 8 – Табло з показаннями лічильника енергоспоживання.

4) перехід від RGB до GREY

Метод cv2.cvtColor() з аргументами у виді image (вхідне зображення) та cv2.COLOR_BGR2GRAY (заміна RGB на відтінки сірого) дозволяє

перевести зображення в необхідний нам формат, який найчастіше використовується (рисунок 9).

Однак в даному випадку, зручніше було використовувати рівні голубого чи зеленого кольору в якості відтінків сірого, тобто image[:,:,0] чи image[:,:,1]. Це дозволяє в нашому випадку отримати сіре зображення без засвічених через червоний колір областей навколо останньої цифри.

Рисунок 9 – Табло, що представлено у відтінках сірого.

5) збільшення контрастності зображення

Для того, щоб повисити якість роботи наступних пунктів, у зображення слід повисити контрастність. Контраст — міра виявлення (і, отже, розпізнавання) об'єкта на якому-небудь тлі. Розрахувати контраст можна, якщо відома яскравість об'єкта і фону, на якому ми спостерігаємо об'єкт:

$C=(Lo-Lф)/Lф$,

де: Lo — яскравість об'єкта; Lф — яскравість фону.

Правильний розрахунок контрасту необхідний для отримання гармонійного розподілу яскравості. Зміну контрасту можна виконати завдяки методу cv2.convertScaleAbs(image, alpha, beta), де alpha - контраст між освітлюваним об'єктом і його найближчим оточенням, beta – яскравість зображення. Результат роботи зображено на рисунку 10.

Рисунок 10 – Табло, рівень контрасту якого було збільшено.

6) згладжені зображення

Для того, аби зменшити кількість шуму, згладити краї та кількість випадкових пікселів перед бінарізацією зображення, має сенс використовувати методи згладжування зображення. В цій роботі необхідну функцію виконує двостороння фільтрація, якою можна скористатися з допомогою методу cv2.bilateralFilter(), параметрами якого є:

- вхідне зображення;
- радіус охвату сусідніх пікселів, що будуть фільтровані відносно центрального пікселя;
- границя різниці між кольорами сусідніх пікселів, починаючи з якої подальші пікселі не будуть використовуватися у фільтрації;
- інтенсивність фільтрування.

Цей фільтр відрізняється високою ефективністю при усуненні шуму, зберігаючи гострі краї. Але робота відбувається повільніше порівняно з іншими фільтрами. Основа двосторонньої фільтрації - Фільтр Гауса, який бере околиці навколо пікселя і знаходить його середнє значення в Гаусі. Фільтр Гауса є функцією простору, тобто під час фільтрації враховуються пікселі, що знаходяться поблизу. Проте не враховується, чи мають пікселі майже однакову інтенсивність. Через це на виході ми маємо розмиті краї.

В той же час двостороння фільтрація використовує дві функції Гауса: в просторі та різниці інтенсивності пікселів. Гаусова функція простору забезпечує розмивання лише пікселів, що знаходяться поблизу, тоді як Гаусова функція різниці інтенсивності гарантує, що для розмивання вважаються лише ті пікселі, що мають інтенсивність, подібні до центрального пікселя. Таким чином, він зберігає ребра, оскільки пікселі на краях будуть відрізнятись великою інтенсивністю. В результаті виконання отримується зображення, приклад якого зображено на рисунку 11.

Рисунок 11 – Результат виконання фільтрації зображення.

7) бінарізація

Бінарізація зображення – перехід представлення зображення у відтінках сірого до використанням двох кольорів (зазвичай чорного та білого). Поширено використовують два види бінарізації – порогову та адаптивну.

Порогова бінарізація для кожного пікселя застосовує однакове критичне значення. Якщо значення пікселя менше цього значення, воно встановлюється на 0, інакше воно встановлюється на максимальне значення , що у відтінках сірого становить 255 (рисунок 12). Функція cv.threshold використовується для застосування порогового значення. Перший аргумент - це вхідне зображення, яке має бути зображенням сірого спектру. Другий аргумент - це порогове значення, яке використовується для класифікації вихідних значень пікселів. Третій аргумент - це максимальне значення, яке присвоюється пікселям, значення яких перевищує поріг. OpenCV забезпечує різні типи порогового значення, яке задається четвертим параметром функції. Базовий поріг, який описано вище, здійснюється за допомогою типу cv2.THRESH_BINARY. cv2.THRESH_BINARY_INV виконує ту ж процедуру, інвертуючи максимальне та мінімальне значення.

Адаптивну бінарізацію доцільно використовувати, якщо зображення має різні умови освітлення в різних областях. Алгоритм визначає поріг пікселя на основі невеликої області навколо нього. Таким чином, отримуються різні пороги для різних областей одного зображення, що дає кращі результати для зображень з неоднорідною освітленістю. Проте для

таких результатів необхідно досить точно та націлено підбирати подальші дії.

Завдяки тому, що під час переводу зображення у відтінки сірого, освітленість було зведено до приближених рівнів, зручніше використовувати порогову бінарізацію зображення. Це дозволяє в подальшому використовувати суму значень пікселів на їх проекціях для знаходження необхідних нам частин зображення.

Рисунок 12 – Представлення зображення в чорному та білому кольорах.

3.3 Алгоритм знаходження табло

На вхід подається бінарізоване зображення. Для того, щоб виділити табло на зображенні, може бути використано один з двох підходів: знаходження контурів чи використання суми пікселів на їх проекції.

Беручи до уваги, що на даному лічильнику циферблат виділено досить чітко завдяки його темному кольору на фоні світлих кольорів лічильника, зручно використати другий підхід, який описано у пункті 3.4. з різницею, що ми послідовно шукаємо по горизонталі та по вертикалі не максимальні екстремуми, а мінімальні, адже перші зустрічі саме з ними свідчать про межі циферблату на зображенні. Результат роботи алгоритму зображено на рисунку 13.

Рисунок 13 – Виокремлене табло лічильника.

3.4 Алгоритм сегментації цифр на циферблаті

На вхід подається отримане перед цим зображення циферблату. Необхідно виокремити кожну цифру. Результат цього кроку, який використовується в якості вхідних даних у наступному етапі, має величезне значення. Топологічний структурний аналіз оцифрованих бінарних зображень за межею проходження простіший до реалізації, адже на ньому побудована функція cv2.findContours(). Проте, цей варіант використано в роботі, адже результат його роботи складніше передбачити, в порівнянні з попереднім, через що він не такий гнучкий в подальшому налаштуванні та відлагодженні програми.

Тому було обрано сегментацію завдяки використанню суми пікселів на їх проекції.

Сума пікселів на їх проекції використовується для знаходження меж зі всіх сторін кожного символу (рисунок 14). Виконуємо горизонтальну проекцію, щоб знайти верхню і нижню границі символів. Сума пікселів на їх проекції - це сума білих пікселів уздовж певної лінії в горизонтальному чи вертикальному напрямках. Коли всі значення вздовж усіх ліній у горизонтальному напрямку обчислюються, отримується сума горизонтальної проекції. Потім граничні значення сум використовуються як пороги для отримання сум вертикальних проекції та визначення за допомогою них меж символу зліва та справа.

Рисунок 14 – Табло лічильника з позначеними межами цифр.

3.5 Алгоритм розпізнавання цифр

Етап розпізнавання є останнім кроком у розробці системи автоматичного розпізнавання показань лічильника енергоспоживання. Таким

чином, він закриває всі процеси, що виконуються для отримання зображення, розташування табло та сегментації. Розпізнавання повинно здійснюватись із зображень, отриманих наприкінці фази сегментації.

3.5.1 Розпізнавання нейронною мережею

Модель навчання, яка використовується для розпізнавання, повинна вміти читати зображення та на виході мати відповідну цифру.

При перших спробах реалізації моделі розпізнавання, було прийняте рішення використовувати якомога простішу модель, а саме повнозв'язний багатошаровий персептрон (MLP). Він надавав доволі точні результати під час тестування програми до часу, коли було використано зображення, на яких були дефекти або числа було видно не повністю чи зміщено. Остання цифра циферблату не є необхідною для надання показань і її можна ігнорувати, адже вона має найбільший шанс опинитися в такому стані. Проте, не зважаючи на це, можна також зустрітися з ситуацією, коли не всі цифри будуть розташовані по центру. Їх підготовка до розпізнавання завдяки сумі значень пікселів на їх проекції та додавання рамок мінімізує можливість таких випадків, проте, водночас, не завжди є можливість використання саме цього алгоритму підготовки зображення.

Тож для того, аби зробити кінцевий програмний компонент максимально гнучким, було прийняте рішення використовувати згорткову нейронну мережу.

3.5.2 Створення моделі

За створення моделі в програмі відповідає функція mnist_cnn_model().

Тип моделі, що використано – Sequential() (послідовність). Послідовність - це найпростіший спосіб побудови моделі в Керасі. Вона дозволяє будувати модель пошарово. Для додавання шарів до нашої моделі використовується функцію model.add().

Модель створена на класичних для згорткової мережі шарах Convolution, Pooling, Flatten, Full-Connected, що позначені класами conv2d, max_pooling2d, flatten, dense, відповідно (рисунок 15).

Мережа на першому шарі приймає на вхід тензори з формою (висота_зображення, ширина_зображення, кількість_каналів). Це форма кожного вхідного зображення: 28 пікселів, 28 пікселів, 1, відповідно, де остання “1” означає, що зображення має відтінки сірого. В коді це позначається в класі Conv2D як параметр input_shape.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 64)                200768
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 257,162
Trainable params: 257,162
Non-trainable params: 0
```

Рисунок 15 – Структура побудованої моделі згорткової нейронної мережі.

В Conv2D шарах нашої моделі кількості вузлів дорівнює 32, 64 та 64, відповідно. Ці числа можуть бути відрегульовано таким чином, щоб вони були більше або менше, залежно від розміру набору даних. У нашому випадку 32, 64 та 64 працюють добре, через що їх і було взято в кінцеву модель.

Розмір ядра - це розмір матриці фільтра для згортки. Отже, розмір ядра 3 означає, що ми матимемо матрицю фільтрів 3x3. В коді це позначається в класі Conv2D як параметр kernel_size.

Між шарами Conv2D і повнозв’язними є шар Flatten. Плоскість служить сполукою між згорткою і щільними шарами. Вона переводить трьохмірний тензор в одномірний, з яким надалі може працювати шар Dense.

Шари Dense – повнозв’язні шари, які використовуються для вихідного шару. Перший Dense шар має розмірність 64, що дорівнює кількості вузлів у останньому Conv2D шарі, завдяки чому можна припустити, що такої кількості достатньо, і, як можна побачити в розділі 3.5.3, це допущення не помилкове. А вихідний шар побудовано з 10 вузлів, по одному на кожен можливий результат (0–9).

Активація - це функція активації шару. В коді позначається в класах Conv2D та Dense як параметр activation. Функція активації, яка використовується для Conv2D шарів та першого Dense шару - це ReLU або випрямлена лінійна активація. В останньому шарі використано Softmax. Ця функція активації робить так, що значення вузлів на виходах знаходяться в діапазоні від 0 до 1, тому висновок може бути інтерпретоване як імовірність. Модель робить своє передбачення на основі того, який варіант має найбільшу ймовірність.

Після побудови модель компілюється. Для компіляції використовуються три параметри: optimizer (оптимізатор), loss (втрати) та metrics (показники).

В проєкті використовується оптимізатор Adam. Його було використано у багатьох прикладах та випадках, з якими я ознайомився під час вивчення нейронних мереж, і його рекомендують як хороший оптимізатор. Оптимізатор Adam регулює швидкість та рівень навчання протягом тренування. Швидкість навчання визначає, наскільки швидко обчислюються оптимальні ваги для моделі. Менша швидкість навчання може призвести до більш точних ваг (до певного моменту), але час, необхідний для обчислення ваг, буде довшим.

Для функції втрат використовується "categorical_crossentropy". Це найпоширеніший вибір для класифікації. Оцінює відсоток, помножений на

0.01, невірно визначених тренувальних прикладів. Нижча оцінка вказує на те, що модель працює краще.

Для ще простішого інтерпретування, використовується метрика «accuracy», яка дозволяє бачити показник точності набору валідації при тренуванні моделі.

3.5.3 Навчання моделі

Для навчання (тренування) створеної моделі викликається функція mnist_cnn_train(model).

Спочатку набір MNIST для тренування та тестування моделі загружається та організовується для подальшого використання. Після чого використовується метод model.fit() з такими параметрами: дані навчання; цільові дані; кількість епох; кількість тренувальних прикладів, що задіється за одну ітерацію; дані для валідації.

Результат роботи методу model.fit(train_data, train_labels_cat, epochs=8, batch_size=64, validation_data=(val_data, val_labels_cat)) зображено на рисунку 16, де train_data, train_labels_cat, val_data та val_labels_cat – попередньо сформовані дані для навчання та тестування на основі MNIST.

Рисунок 16 – результат навчання та тестування моделі.

Після навчання моделі вона зберігається методом model.save(path_to_model), де path_to_model – шлях до файлу, в якому буде збережено модель. Зазвичай, файл має формат “file_name.h5”, де file_name - бажане ім’я файлу.

3.5.4 Використання моделі для класифікації

Для використання моделі для класифікації об’єкту на зображенні, насамперед модель завантажується методом tf.keras.models.load_model(path_to_model) та у змінну model записується шлях до цього об’єкту.

Наступним кроком викликається функція cnn_digits_predict(model, path_to_image), де path_to_image – це шлях до зображення, яке буде класифікуватися.

Загальний вигляд в коді функції:

```
def cnn_digits_predict(model, image_file):
    image_size = 28
    img = keras.preprocessing.image.load_img(image_file,
target_size=(image_size, image_size), color_mode='grayscale')
    img_arr = np.expand_dims(img, axis=0)
    img_arr = 1 - img_arr/255.0
    img_arr = img_arr.reshape((1, image_size, image_size, 1))
    result = model.predict_classes([img_arr])
    return result[0]
```

Функція виконую таку послідовність дій:

1) завантажує зображення та трансформує його до заданого розміру target_size та режиму кольору color_mode методом keras.preprocessing.image.load_img(image_file, target_size, color_mode);

2) приводить завантажене зображення до стандартного np.array;
3) викликає метод model.predict_classes(), що використовує модель нейронної мережі для класифікації зображення значення, що відповідає об’єкту (цифрі), який треба було розпізнати на зображенні.

3.6 Алгоритми перевірки відповідей та їх запису

Зчитування та запис у файл виконуються з використанням контекстного менеджеру with, що автоматично зачинить файл після виконання позначених у ньому дій, та стандартної функції open().

Перетворення даних з CSV формату у списки та навпаки виконується завдяки бібліотечним методам csv.reader() та csv.writer().

Перевірка отриманих відповідей виконується функцією result_checking(inputDigitsStr, CSVFile, deltaMin, deltaMax), де:

- inputDigitsStr – вхідний рядок з розпізнаних цифр показань;
- CSVFile – таблиця, з якої беруться дані для перевірки;
- deltaMin – мінімальне значення різниці поточного та останнього показань, deltaMax – мінімальне значення різниці показань.

Принцип перевірки побудовано на послідовності таких дій:

1) перевірка кількості отриманих цифр. Якщо вона менша за 6, функція поверне:
   "Current readings { inputDigitsStr } length is less than 6";
2) зчитування таблиці з файлу CSVFile, в якому зберігаються попередні показання лічильника та запис останньої строки таблиці у список lastRow;
3) знаходження різниці delta між останніми показаннями та отриманими показаннями;
4) перевірка, чи входить різниця у рамки використання електроенергії (описано у розділі 1.2.1):

a) перевірка, чи delta менше або дорівнює нулю. Якщо істина, функція повертає строку:

"Current readings "{inputDigitsStr}" are less than last "{lastRow[0]}"";

b) перевірка, чи не менша delta за deltaMin. Якщо менша, функція повертає строку:

"Current readings "{inputDigitsStr}" are less than the minimum expected "{lastRow[0] + deltaMin}" after last "{lastRow[0]}"";

c) перевірка, чи не більша delta за deltaMax. Якщо більша, функція повертає строку:

"Current readings "{inputDigitsStr}" are greater than the maximum expected "{lastRow[0] + deltaMax}" after last "{lastRow[0]}"".

5) якщо не було повернуто одного з попередніх значень, функція повертає строку: "OK".

Запис даних виконується функцією result_writing(inputDigitsStr, inputDigitsMeterFilename, CSVFile), де

- inputDigitsStr – вхідний рядок з розпізнаних цифр;
- inputDigitsMeterFilename – шлях до зображення, з якого було виконано розпізнавання;
- CSVFile – таблиця, до якої буде записано отримані дані.

Запис виконується за алгоритмом:

1. для того, щоб не втратити попередні дані з таблиці, вони зчитуються та записуються в output;
2. у отриманій змінній inputDigitsStr символи, чий порядковий номер більше “6”, виокремлюються крапкою;
3. формується список даних (показання лічильника, назва файлу, з якого було отримано результат, дата запису показань), що буде додаватися до таблиці:

data = [inputDigitsStr, inputDigitsMeterFilename, str(datetime.now())], де datetime.now() – функція стандартної бібліотеки datetime, що повертає поточну дату;

4. додавання сформованого списку data до попередніх даних output;
5. запис даних зі змінних output до файлу CSVFile.

Звернення до цих функції виконуються у вигляді:

```
resultChecking = result_checking(predictionResult, CSVFile, deltaMin=30, deltaMax=50)
if resultChecking == "OK":
    result_writing(predictionResult, meterJPG, CSVFile)
else:
    print(resultChecking)
```

Тобто дані записуються до таблиці тільки після того, як перевірка розпізнаних показань повертає строку “ОК”, а інакше в термінал виводиться рядок відповіді функції result_checking().

3.7 Прибирання залишків роботи програми

Для того, аби забезпечити більш зручне використання та відлагодження програми, використовуються буферні записи зображень цифр до файлів, що залишаються після її роботи. Звісно, їх треба прибрати. Для цього використовується модуль OS. Модуль надає багато функції для роботи з операційною системою, при чому їх поведінка, зазвичай, не залежить від операційної системи, на якій їх буде використано. Тож він не позбавить програмне рішення кросплатформності.

```
path=os.path.join(os.path.abspath(os.path.dirname(__file__)),
filename)
os.remove(path)
```

У першому рядку формується та запам’ятовується шлях до файлу, у другій – файл, що знаходиться за попередньо визначеним шляхом, видаляється. Додаючи ці дії в цикл, в якому ми зберігаємо отриманні після сегментації цифри у файли та викликаємо функцію їх розпізнавання, ми отримуємо такий кінцевий його вигляд

```
digits = image_preparing(imageFile)[0]
predictionResult = ""
num = 0
for i in digits:
    filename = "{}.jpg".format(num)
    cv2.imwrite(filename, I)
    currentDigit = cnn_digits_predict(model, filename)
    predictionResult += str(currentDigit)
    path = os.path.join(os.path.abspath(os.path.dirname(__file__)),
filename)
    os.remove(path)
    num = num + 1
```

3.8 Висновки до розділу

Беручи до уваги те, що програмний компонент створюється окремо від цільового проєкту, а також на наявність багатьох змінних у вхідному зображенню, доцільніше всього корегувати роботу програми в залежності від кожного випадку. Деякі інструменти, що можуть бути використані в алгоритмах для вирішення цієї задачі в інших умовах, згадано в розділі, проте

не продемонстровано чи не описано. Причиною цього є те, що вони чи не є оптимальними або потрібними в умовах, що використовується в проєкті, чи мають досить багато варіацій налаштування, через що їх опис зайняв би багато місця, чи дають такі ж самі результати, що й використані засоби. Ознайомитись з їх описами можна в документаціях до них, які вказано в використаних джерелах.

Завдяки тому, що рішення поставленої задачі досить послідовне та має небагато логічних блоків, програма не потребує додаткового аналізу патернів проєктування.

Звісно, ідеальним було би використання одного потужного алгоритму, що зміг би охопити всі можливі випадки. Проте для виконання цієї задачі необхідно розробити занадто складні та важкі програмі засоби, що на мою думку не будуть більш раціональним рішенням, ніж використання сучасних лічильників та системи АСКОЕ.

4 АНАЛІЗ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

4.1 Тестування системи

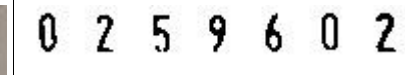
Для першого етапу тестування програного засобу використано додаткову функцію, що об’єднує сегментовані частини зображення для простішої демонстрації їх у таблиці:

```
totalDigits = digits[0]
    for i in range(1, len(digits) – 1):
        totalDigits = np.concatenate((totalDigits, digits[i]), axis=1)
cv2.imwrite('out.jpg', totalDigits)
```

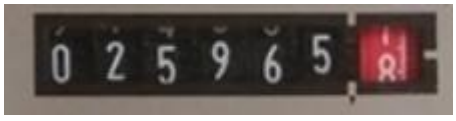
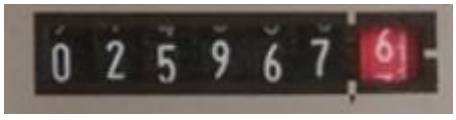
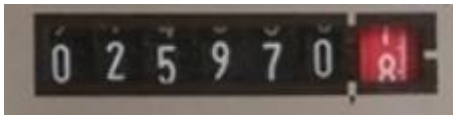
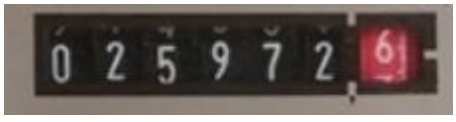
Під час цього етапу тестування перевіряється робота двох компонентів програми: сегментації та розпізнавання. Результати тестування надаються в таблиці 1 зі стовпчиками:

- “Зображення”, що вміщує в себе вхідне зображення;
- “Сегментація”, що вміщує в себе результат роботи алгоритму сегментації та виділення цифр для відповідного вхідного зображення;
- “Розпізнавання”, що вміщує в себе відповідний результат роботи алгоритму розпізнавання на основі результатів, отриманих при виконанні алгоритму сегментації.

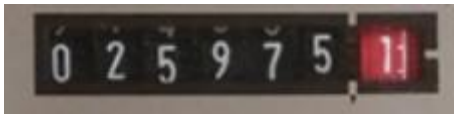
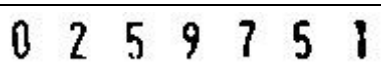
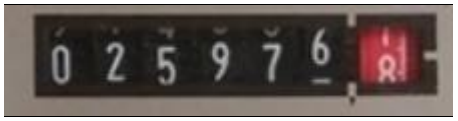
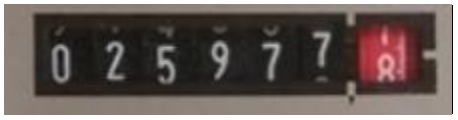
Таблиця 1 – Результати тестування основних компонентів програми

Зображення	Сегментація	Розпізнавання
		025960,2

Продовження таблиці 1

Зображення	Сегментація	Розпізнавання
	0 2 5 9 6 1 8	025961,8
	0 2 5 9 6 2 8	025962,2
	0 2 5 9 6 3 2	025963,2
	0 2 5 9 6 4 8	025964,8
	0 2 5 9 6 5 8	025965,8
	0 2 5 9 6 6 1	025966,1
	0 2 5 9 6 7 6	025967,6
	0 2 5 9 6 8 8	025968,7
	0 2 5 9 6 9 8	025969,8
	0 2 5 9 7 0 8	025970,8
	0 2 5 9 7 2 6	025972,6

Продовження таблиці 1

Зображення	Сегментація	Розпізнавання
		025975,1
		025976,8
		025977,8

Другим етапом виконується тестування функції перевірки повернених значень функцій result_checking(). Результат тестування продемонстровано в таблиці 2. Вхідні

Таблиця 2 – Результати тестування функції result_checking()

Останні показання у таблиці	Поточні показання	deltaMin, deltaMax	Значення, що повернула функція
025972.6	025969.8	1, 1	Current readings “025969” are less than last “025972”
025972.6	025977.8	1, 2	Current readings “025977” are greater than the maximum expected “25974” after last “025972”
025970.8	025972.6	3, 5	Current readings “025972” are less than the minimum expected “25973” after last “025970”

4.2 Аналіз отриманих результатів тестування

Як можна побачити з таблиці 1, алгоритм впевнено почуває себе при обробці та розпізнаванні вхідних зображень, виконаних в однакових умовах, адже алгоритм був відповідно налаштований та використовується нейронна мережа, що підходить для вирішення поставленої задачі.

Беручи до уваги, що перші 6 цифр на табло, які потребують найбільшої уваги, мають відносно чіткий хід і не буває випадків, коли цифру видно не повністю, можна бути впевненим, що в кожному випадку алгоритм буде надавати достовірний результат.

Що ж стосується сьомої цифри, то у випадках, коли видно її важливі ознакові частини, її скоріш за все буде класифікована вірно, а інакше – великий шанс на помилку. Проте вона не є функціонально чи безпосередньо необхідною, тож нема необхідності в перетворенні коду.

Отримані ж результати, що наведено в таблиці 2, свідчать про те, що алгоритм має підтримку в якості додаткових перевірок отриманих значень та, щонайменше, не добавить користувачеві проблем з постачальником енергії. Окрім цього, він також повідомить про наявність помилки і виведе інформацію щодо неї.

4.3 Рекомендації щодо використання розробки

Розробка найбільш досконало виглядає в системі, як описувалося в розділі 1.3, що побудована на статично розташованій камері, модулів відправки та отримання знімків та програмному засобі, що було розроблено в даному проєкті.

Важливим є те, що характеристики вхідних зображень при різних умовах використання системи, зазначеної вище, також будуть різними. Тож програмний засіб потребує налаштування при суттєвих змінах у вхідних зображеннях.

Налаштування виконуються інтуїтивно зрозумілими змінами у програмному коді та стосуються саме підготовки зображення до розпізнавання.

Таблиця 3 демонструє місця в функції image_preparing(), що потребуватимуть налаштування, та зміни, до яких це призводитиме.

Таблиця 3 – Приклади налаштування блоку обробки зображення

Початкове зображення	Зміни	Отримане зображення
	Resize: image_prepare.py, line 11. size = 800 to size = 600	
	Grayscale: image_prepare.py, line 21. imageCut[:,:,1] to imageCut[:,:,2]	
	Contrast.beta: image_prepare.py, line 25. beta = -100 to beta = 0	
	Contrast.alpha: image_prepare.py, line 25. beta = -100 to beta = 0 and alpha=3.0 to alpha=0	

Продовження таблиці 3

Початкове зображення	Зміни	Отримане зображення
	Filter: image_prepare.py, line 29. sigmaColor = 40 to sigmaColor = 100	
	Threshold: image_prepare.py, line 34. thresh = 200 to thresh = 100	

4.4 Рекомендації щодо подальшого вдосконалення

Для вдосконалення програмного засобу, доцільним може бути:

1) використання додаткової нейронної мережі, яка зможе знаходити циферблат на зображенні. Це дозволить використовувати засіб з мобільною камерою в якості веб чи мобільного додатку. Проте, для реалізації цього рішення потрібно зібрати достатньо великий набір даних та побудувати підходящу нейронну модель, що дозволить бути впевненим в надійності засобу;

2) використання додаткової нейронної мережі, яка зможе ідентифікувати лічильник. Це дозволить розширити сферу використання програмного засобу, поліпшеного в попередньому пункті, для розпізнавання, по-перше, всіх видів лічильників споживання електроенергії, а по-друге – інших видів лічильників (споживання газу, води, тощо). Плюсом такого підходу є те, що не потрібно налаштовувати програмний засіб для кожного з лічильників, а також автоматизує рішення щодо використання даних, без втручання людини.

3) додавання інтерфейсу користувача. Це вдосконалення має сенс саме при розширенні програмного засобу, адже його суть полягає в тому, щоб надати користувачеві зручності та інформативності у вистежуванні роботи засобу;

4) використання нейронної мережі, що буде навчена корегувати зображення та максимально точно його сегментувати. Це замінить алгоритми, що описані у розділі 3.2.

4.5 Висновок до розділу

Тестування роботи основних компонентів програми пройшло успішно та надало очікувані результати в таблиці 1. Однак, зважаючи на те, що вибірка для тестування досить невелика (хоч і покриває більшість критичних меж), не можна бути до кінця впевненим, що нейронна мережа не помилиться в якомусь непередбачуваному випадку.

Для мінімізації можливих наслідків помилок, було добавлено «контролера», що дозволяє фіксувати їх. Ефективність та принципи його роботи продемонстровано в таблиці 2.

Для аналізу результатів покрокових дій в алгоритмі обробки зображення, було сформовано таблицю 3. Вона надає можливість наглядно побачити та зрозуміти, які налаштування слід використовувати в ти чи інших випадках та яких результатів очікувати.

На останок, важливим було проведення аналізу розробленого програмного засобу та виявлення шляхів його розвитку. Це дозволяє побачити, що він, окрім того, що є вже готовим продуктом, може бути суттєво вдосконалений, що робить його розробку ще значущою, ніж у випадку, якщо б дана реалізація була «піком» можливостей.

ВИСНОВКИ

Метою даного дипломного проєкту було створення програмного засобу для розпізнавання показань лічильника енергоспоживання.

Перед виконанням завдання було проведено дослідження предметної області, розглянуто проблеми користувачів, проаналізовано існуючі системи та програмні засоби, встановлено актуальність розробки, сформовано вимоги до програмного забезпечення та знайдено можливі способи вирішення поставленої задачі.

Під час виконання дипломного проєкту були занотовані проведені дослідження та аналізи, а також розроблено та протестовано програмний засіб, що відповідає вимогам до нього та вирішує поставлені проблеми.

Пояснювальна записка містить у собі змістовне обґрунтування вибору технологій розробки, мови програмування, бібліотек на фреймворків. Крім цього, надано опис використаних методів та розроблених алгоритмів, моделей машинного навчання, опис архітектури програми, приклади її роботи, рекомендації для використання та подальшого розроблення.

Розроблений програмний засіб здатен розпізнавати показання лічильника енергоспоживання з фотографії, записувати їх та порівнювати з попередніми показаннями. Він може бути використана для автоматизації процесу фіксування та надання показань лічильника.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Передача показань лічильника. Офіційний сайт ДТЕК : URL : https://www.dtek-kem.com.ua/ua/meter (дата звернення 23.12.19)

2. Усе про лічильники електроенергії. Офіційний сайт ДТЕК: URL : https://www.dtek-kem.com.ua/ua/metering-devices (дата звернення 25.12.19)

3. Технічний зір : URL : http://wiki.technicalvision.ru/index.php/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0 (дата звернення 05.01.20)

4. Методи розпізнавання тексту : URL : https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8_%D1%80%D0%BE%D0%B7%D0%BF%D1%96%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D1%82%D0%B5%D0%BA%D1%81%D1%82%D1%83 (дата звернення 10.01.20)

5. Introduction to OCR. Towards Data Science : URL : https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa (дата звернення 24.01.20)

6. Advantages and Disadvantages of Python Programming Language : URL : https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121 (дата звернення 02.02.20)

7. Python 3 documentation: URL : https://docs.python.org/3/ (дата звернення 06.02.20)

8. 10 Python image manipulations tools. Towards Data Science : URL : https://towardsdatascience.com/image-manipulation-tools-for-python-6eb0908ed61f (дата звернення 11.02)

9. Numpy manual : URL : https://numpy.org/doc/1.18 (дата звернення 12.02)

10. OpenCV-Python Tutorials: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html (дата звернення 15.02)

11. OpenCV User Guide – OpenCV 2.4.13.7 documentation: URL : https://docs.opencv.org/2.4/doc/user_guide/user_guide.html (дата звернення 15.02)

12. Datasets: URL: https://keras.io/api/datasets/ (дата звернення 20.02)

13. Yann LeCun, Corinna Cortes and Christopher J.C. Burges. MNIST handwritten digit database: URL : http://yann.lecun.com/exdb/mnist/ (дата звернення 20.02)

14. Complete Guide to Artifical Neural Network Concepts & Models: URL : https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/#perceptrons (дата звернення 21.02)

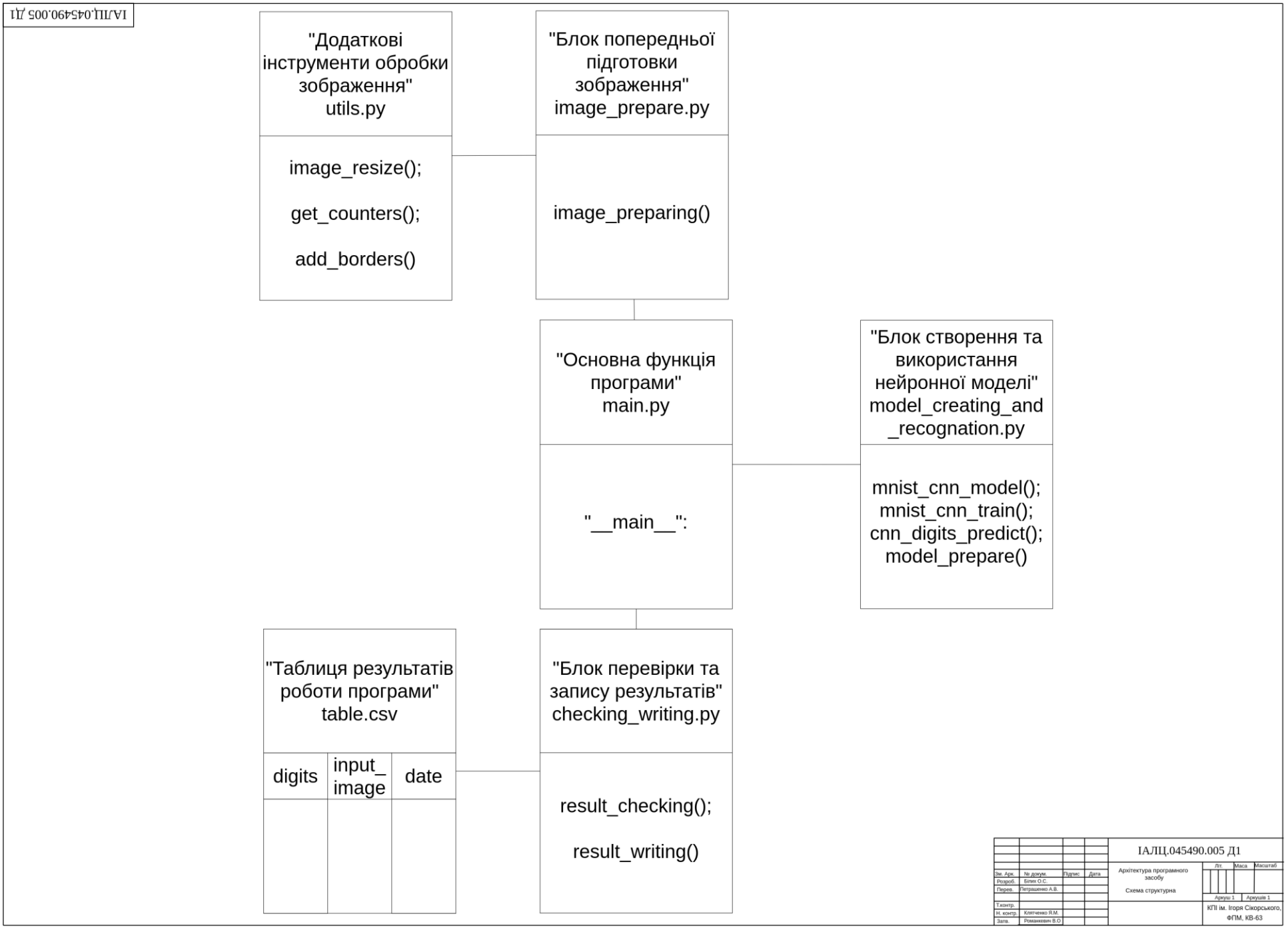
15. TensorFlow documentation : URL : https://www.tensorflow.org/guide?hl=ru (дата звернення 24.02)

16. Keras Developer guides : URL : https://keras.io/guides/ (дата звернення 26.02)

17. Python Convolutional Neural Network: Creating a CNN in Keras, TensorFlow and Plain Python: URL : https://missinglink.ai/guides/convolutional-neural-networks/python-convolutional-neural-network-creating-cnn-keras-tensorflow-plain-python/ (дата звернення 01.03)

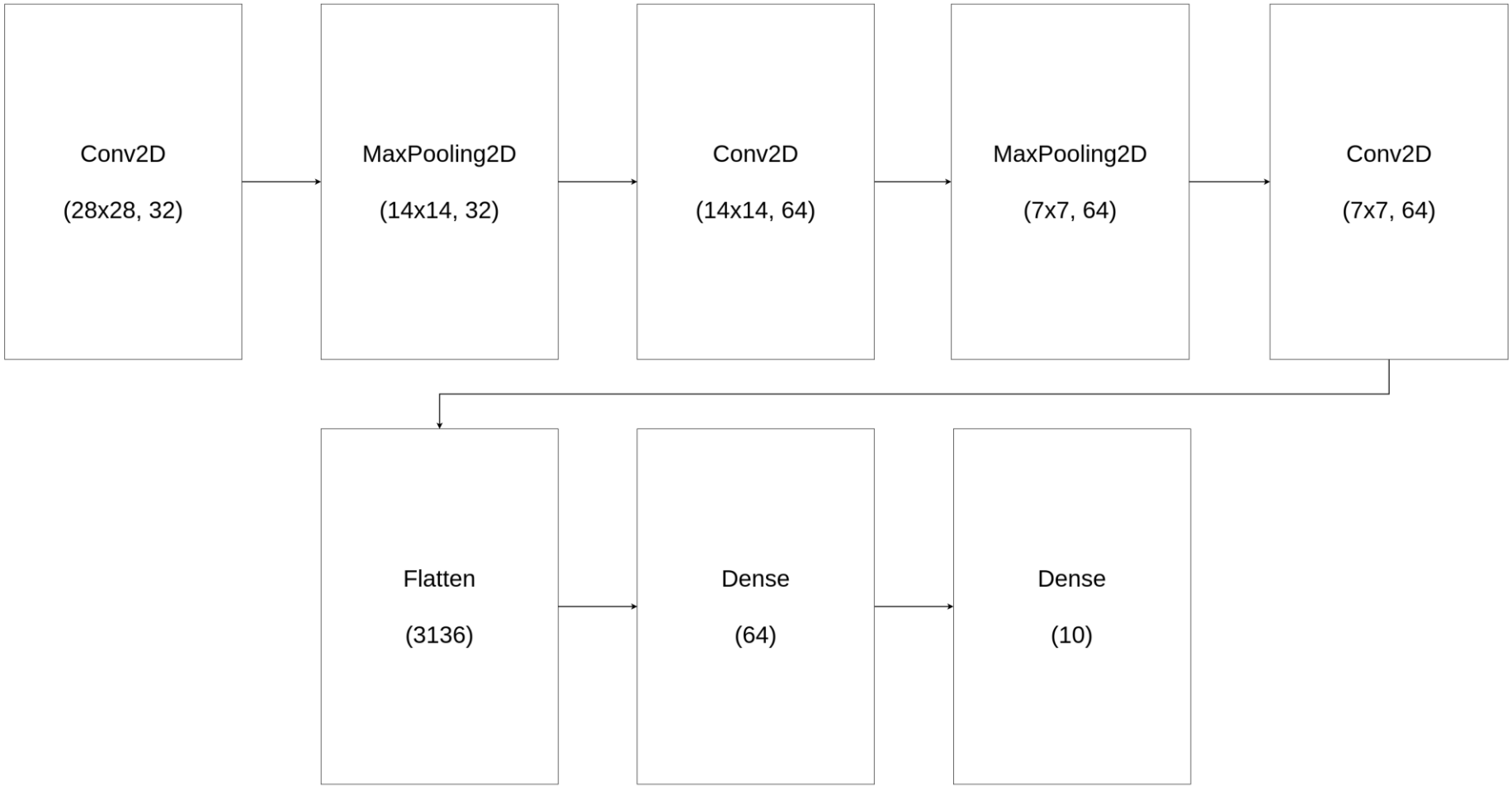
ІАЛЦ.045490.005 Д1

ДОДАТОК А



						ІАЛЦ.045490.005 Д1	
						Архітектура програмного засобу	Літ. / Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Білих О.С.				Схема структурна	
Перев.		Петрашенко А.В.					Аркуш 1 / Аркушів 1
Т.контр.							
Н. контр.		Клятченко Я.М.					КПІ ім. Ігоря Сікорського, ФІПМ, КВ-63
Затв.		Романкевич В.О					

ДОДАТОК Б

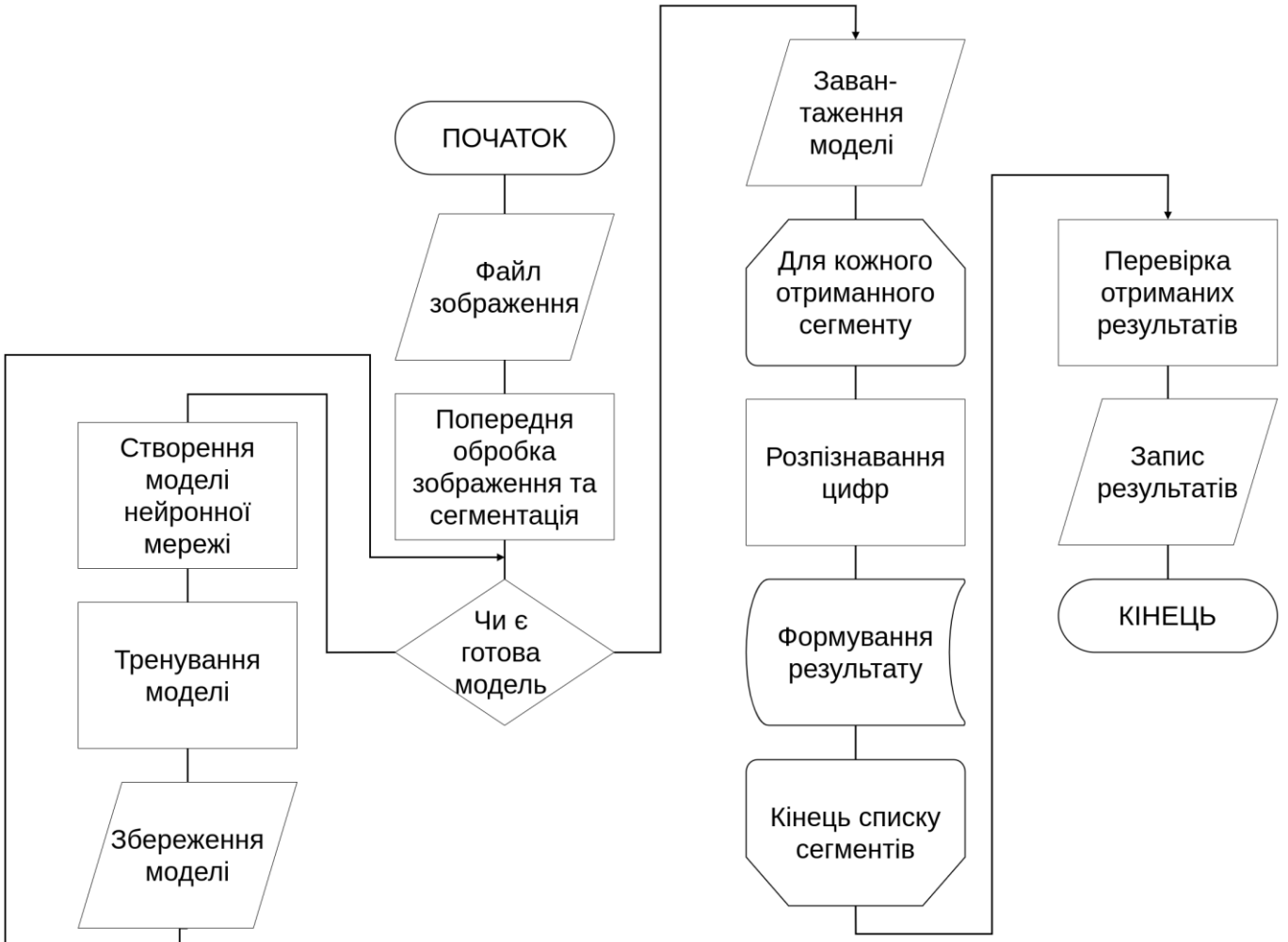


				ІАЛЦ.045490.006 Д2		
Зм. Арк.	№ докум.	Підпис	Дата	Структура нейронної мережі	Літ.	Маса / Масштаб
Розроб.	Білик О.С.			Схема структурна		
Перев.	Петрашенко А.В.				Аркуш 1	Аркушів 1
Т.контр.					КПІ ім. Ігоря Сікорського, ФПМ, КВ-63	
Н. контр.	Клятченко Я.М.					
Затв.	Романкевич В.О					

ДОДАТОК В

				ІАЛЦ.045490.007 Д3		
				Послідовність підготовки зображення перед розпізнаванням Схема алгоритму	Літ.	Маса	Масштаб
Зм. Арк.	№ докум.	Підпис	Дата				
Розроб.	Білих О.С.				Аркуш 1	Аркушів 1
Перев.	Петращенко А.В.					
Т.контр.					КПІ ім. Ігоря Сікорського, ФПМ, КВ-63	
Н. контр.	Клятченко Я.М.					
Затв.	Романкевич В.О					

ДОДАТОК Г



				ІАЛЦ.045490.008 Д4	
Зм. Арк.	№ докум.	Підпис	Дата	Алгоритм розпізнавання цифр	Лт. / Маса / Масштаб
Розроб.	Білик О.С.			Схема алгоритму	
Перев.	Петрашенко А.В.				Аркуш 1 / Аркушів 1
Т.контр.					КПІ ім. Ігоря Сікорського, ФПМ, КВ-63
Н. контр.	Клятченко Я.М.				
Затв.	Романкевич В.О				

ДОДАТОК Д

ЛІСТИНГ ПРОГРАМИ

main.py

```
from image_prepare import *
from utils import *
from model_creating_and_recogntaion import *
from checking_writing import *
import os

if __name__ == "__main__":

    modelPath = 'cnn_digits_28x28_2.h5'
    # model_prepare(modelPath) ##uncomment to create model
    model = tf.keras.models.load_model(modelPath)
    model.summary()

    n = 14
    imageFile = "img/{}.jpg".format(n)
    CSVFile = "table.csv"

    imagePreparingResult = image_preparing(imageFile, configFlag=0)
    digits = imagePreparingResult[0]
    meter = imagePreparingResult[1]

    meterJPG = "output/{}_{}.jpg".format(n, str(datetime.now().date()) + ":" + str(datetime.now().time()))
    cv2.imwrite(meterJPG, meter)

    predictionResult = ""
```

```
    num = 0
    for i in digits:
        filename = "digit_{}.jpg".format(num)
        cv2.imwrite(filename, i)
        currentDigit = cnn_digits_predict(model, filename)
        predictionResult += str(currentDigit)
        print(currentDigit, " <- digit{}: ".format(num))
        path = os.path.join(os.path.abspath(os.path.dirname(__file__)), filename)
        os.remove(path)
        num = num + 1

    resultChecking = result_checking(predictionResult, CSVFile, deltaMin=1, deltaMax=100)
    if resultChecking == "OK":
        result_writing(predictionResult, meterJPG, CSVFile)
    else:
        print(resultChecking)

    totalDigits = digits[0]
    for i in range(1, len(digits)):
        totalDigits = np.concatenate((totalDigits, digits[i]), axis=1)
    cv2.imwrite('output/{}_out.jpg'.format(n), totalDigits)
```

image_prepare.py

```
import numpy as np
import cv2
from utils import *

def image_preparing(imageFile, configFlag):
    # Read the image file
    image = cv2.imread(imageFile)

    # Resize the image
    size = 800
```

```
imageResized = image_resize(image, size)


# Cutting the required part

imageCut = imageResized[int(0.495*size):int(0.565*size), int(0.32*size):int(0.6*size)]


# RGB to Gray scale conversion

# imageGrayscaled = cv2.cvtColor(imageCut, cv2.COLOR_BGR2GRAY)

imageGrayscaled = imageCut[:,:,1]


# Make image more contrasting

imageContrasted = cv2.convertScaleAbs(imageGrayscaled, alpha=3.0, beta=-100)


# Noise removal with iterative bilateral filter

imageBilfiltred = cv2.bilateralFilter(imageContrasted, 10, 40, 10)


# Threshold the image

ret, imageThresholded = cv2.threshold(imageBilfiltred, 200, 255, cv2.THRESH_BINARY)


if configFlag == 1:

    cv2.imshow("0 - Resized", imageResized)

    cv2.imshow("1 - Cut", imageCut)

    cv2.imshow("2 - Grayscaled", imageGrayscaled)

    cv2.imshow("3 - Contrasted", imageContrasted)

    cv2.imshow("4 - Filtered", imageBilfiltred)

    cv2.imshow("5 - Thresholded", imageThresholded)


# Find counters using projections

counters = get_counters(imageThresholded, configFlag = configFlag)


# Adding borders to found numbers

digits = add_borders(imageContrasted, counters, 28)


# Use softer binarization to reduce data loss

digitsThresholded = []
```

```
    for i in digits:
        ret, tmp = cv2.threshold(i, 220, 255, cv2.THRESH_BINARY_INV)
        digitsThresholded.append(tmp)

    if configFlag == 1:
        cv2.waitKey(0)
    result = [digitsThresholded, imageCut]
    return result
```

checking_writing.py

```
import csv
from datetime import datetime

def result_checking(inputDigitsStr, CSVFile, deltaMin, deltaMax):
    if len(inputDigitsStr) < 6:
            return "Current readings \"{}\" length is less than 6".format(inputDigitsStr)
    with open(CSVFile, 'r') as csvF:
        csvReader = csv.reader(csvF, delimiter=',')
        num = 0
        for row in csvReader:
            lastRow = row
            num += 1
        if num > 1:
            delta = int(inputDigitsStr[0:6]) - int(lastRow[0][0:6])
            if delta <= 0:
                return "Current readings \"{}\" are less than last \"{}\"".format(inputDigitsStr[0:6], lastRow[0][0:6])
            elif delta > deltaMax:
                return "Current readings \"{}\" are greater than the maximum expected \"{}\" after last \"{}\"".format(inputDigitsStr[0:6], str(
int(lastRow[0][0:6]) + deltaMax ), lastRow[0][0:6])
            elif delta < deltaMin:
                return "Current readings \"{}\" are less than the minimum expected \"{}\" after last \"{}\"".format( inputDigitsStr[0:6], str(
int(lastRow[0][0:6]) + deltaMin ), lastRow[0][0:6])
    return "OK"
```

```
def result_writing(inputDigitsStr, inputDigitsMeterFilename, CSVFile):
    output = []
    with open(CSVFile, 'r') as csvF:
        csvReader = csv.reader(csvF, delimiter=',')
        for row in csvReader:
            output.append(row)
    if len(inputDigitsStr) == 6:
        data = [inputDigitsStr, inputDigitsMeterFilename, str(datetime.now())]
    else:
        data = [inputDigitsStr[0:6] + "." + inputDigitsStr[6], inputDigitsMeterFilename, str(datetime.now())]
    output.append(data)
    with open(CSVFile, 'w') as csvF:
        writer = csv.writer(csvF, delimiter=',')
        for lines in output:
            writer.writerow(lines)
```

model_creating_and_recognition.py

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Activation, BatchNormalization, AveragePooling2D
from tensorflow.keras.optimizers import SGD, RMSprop, Adam
import tensorflow_datasets as tfds  # pip install tensorflow-datasets
import tensorflow as tf
import logging
import numpy as np
import time

def mnist_cnn_model():
    image_size = 28
```

```
    num_channels = 1  # 1 for grayscale images
    num_classes = 10  # Number of outputs
    model = Sequential()
    model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu',
            padding='same',
            input_shape=(image_size, image_size, num_channels)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu',
            padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu',
            padding='same'))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(optimizer=Adam(), loss='categorical_crossentropy',
            metrics=['accuracy'])
    return model


def mnist_cnn_train(model):
    (train_digits, train_labels), (test_digits, test_labels) = keras.datasets.mnist.load_data()
    # Get image size
    image_size = 28
    num_channels = 1  # 1 for grayscale images
    # re-shape and re-scale the images data
    train_data = np.reshape(train_digits, (train_digits.shape[0], image_size, image_size, num_channels))
    train_data = train_data.astype('float32') / 255.0
    # encode the labels - we have 10 output classes
    # 3 -> [0 0 0 1 0 0 0 0 0 0], 5 -> [0 0 0 0 0 1 0 0 0 0]
    num_classes = 10
    train_labels_cat = keras.utils.to_categorical(train_labels, num_classes)

    # re-shape and re-scale the images validation data
    val_data = np.reshape(test_digits, (test_digits.shape[0], image_size, image_size, num_channels))
```

```
    val_data = val_data.astype('float32') / 255.0
    # encode the labels - we have 10 output classes
    val_labels_cat = keras.utils.to_categorical(test_labels, num_classes)

    print("Training the network...")
    t_start = time.time()

    # Start training the network
    model.fit(train_data, train_labels_cat, epochs=8, batch_size=64,
        validation_data=(val_data, val_labels_cat))

    print("Done, dT:", time.time() - t_start)
    return model

def cnn_digits_predict(model, image_file):
    image_size = 28
    img = keras.preprocessing.image.load_img(image_file,
target_size=(image_size, image_size), color_mode='grayscale')
    img_arr = np.expand_dims(img, axis=0)
    img_arr = 1 - img_arr/255.0
    img_arr = img_arr.reshape((1, image_size, image_size, 1))
    result = model.predict_classes([img_arr])
    return result[0]

def model_prepare(modelPath):
    model = mnist_cnn_model()
    model = mnist_cnn_train(model)
    model.save(modelPath)
```